

WHAT IS CLAIMED IS:

- 1 1. A method for modifying serial dependencies in a procedure, said method
2 comprising:
3 building a graph representation of said procedure, said graph representation
4 having an origin and including a unique position, relative to said origin, for each
5 memory operation in said procedure;
6 designating a location type for each memory operation in said graph
7 representation; each said location type based on a characteristic of said
8 corresponding memory operation;
9 identifying a first memory operation having the same location type as a
10 second memory operation; wherein said first memory operation is positioned closer
11 to said origin than said second memory operation, and said graph representation
12 does not include any additional memory operations of the same location type
13 between the first and second memory operations; and
14 determining whether said graph representation includes at least one program
15 operation between said first and second memory operations, and when said
16 determination is positive, moving said second memory operation to a new position in
17 said graph representation that is closer to said first memory operation.
- 1 2. The method of claim 1 wherein
2 the building step includes the step of assigning an initial set of serial
3 dependencies between program operations represented in said graph
4 representation; and
5 the moving step includes (i) removing one or more of the serial dependencies
6 in said initial set of serial dependencies that is associated with said second memory
7 operation and (ii) creating a new serial dependency between said first memory
8 operation and said second memory operation.
- 1 3. The method of claim 1 wherein said graph representation is an intermediate
2 representation.

1 4. The method of claim 3 wherein said intermediate representation is a static
2 single assignment graph embedded in a control flow graph.

1 5. The method of claim 1 wherein said moving step results in said second
2 memory operation being advanced in a schedule of machine instructions.

1 6. The method of claim 1 wherein said first memory operation is positioned
2 before a repetitive loop in said procedure and said second memory operation is
3 within said repetitive loop; said graphic representation including a phi node that
4 corresponds to a loop back position in said repetitive loop;

5 said designating step further comprising a step of advancing through said
6 repetitive loop in order to determine a location type for each memory operation in
7 said repetitive loop;

8 wherein

9 when a memory operation having the same location type as said first and
10 second memory operation exists in said loop, said new position in said graph
11 representation is a position that is serially dependent upon said phi node; and

12 when a memory operation having the same location type as said first and
13 second memory operation does not exist in said loop, said new position in said
14 graph representation is a position that is not serially dependent on any operation in
15 the loop.

1 7. The method of claim 1 wherein said first memory operation is a store or an
2 array store and said second memory operation is a load or an array load.

1 8. The method of claim 1 wherein said procedure includes a calling procedure
2 and said first memory operation is in said calling procedure and said second
3 memory operation is in a called procedure that is called by an operation in said
4 calling procedure.

1 9. The method of claim 8 wherein said moving step results in a placement of
2 said second memory operation in said calling procedure.

1 10. The method of claim 1 wherein said location type of each memory operation
2 is selected from the group consisting of a predefined set of base types, a predefined
3 set of array types, object types, and object field types.

1 11. The method of claim 1 wherein the building step further comprises adding a
2 global store dependency to each operation in said procedure that reads a variable
3 from or stores a variable to memory; the method further comprising:

4 generating a schedule of machine instructions in accordance with said graph
5 representation, wherein each said machine instruction in said schedule of machine
6 instructions, which corresponds to an operation that reads a variable from or stores
7 a variable to memory, is ordered in accordance with said global store dependency
8 associated with said operation.

1 12. The method of claim 1 wherein a first operation affects a value of a variable
2 stored in memory and a second operation serially follows said first operation, said
3 building step further comprising adding a global store dependency from said second
4 operation to said first operation; the method further comprising:

5 generating a schedule of machine instructions in accordance with said graph
6 representation, wherein said machine instructions in said schedule of machine
7 instructions corresponding to said second operation are scheduled after said
8 machine instructions corresponding to said first operation.

1 13. A computer program product for use in conjunction with a computer system,
2 the computer program product capable of modifying serial dependencies in a
3 procedure, the computer program product comprising a computer readable storage
4 medium and a computer program mechanism embedded therein, comprising:

5 instructions for building a graph representation of said procedure, said graph
6 representation having an origin and including a unique position, relative to said
7 origin, for each memory operation in said procedure;

8 instructions for designating a location type for each memory operation in said
9 graph representation; each said location type based on a characteristic of said
10 corresponding memory operation;

11 instructions for identifying a first memory operation having the same location
12 type as a second memory operation; wherein said first memory operation is
13 positioned closer to said origin than said second memory operation, and said graph
14 representation does not include any additional memory operations of the same
15 location type between said first and second memory operations; and
16 instructions for determining whether said graph representation includes at
17 least one program operation between said first and second memory operations, and
18 when said determination is positive, for moving the second memory operation to a
19 new position in said graph representation that is closer to said first memory
20 operation.

1 14. The computer program product of claim 13 wherein:
2 the instructions for building include instructions for assigning an initial set of
3 serial dependencies between program operations represented in the graph; and
4 the instructions for moving include (i) instructions for removing one or more of
5 the serial dependencies in said initial set of serial dependencies that is associated
6 with said second memory operation and (ii) creating a new serial dependency
7 between said first memory operation and said second memory operation.

1 15. The computer program product of claim 13 wherein said graph representation
2 is an intermediate representation.

1 16. The computer program product of claim 15 wherein said intermediate
2 representation is a static single assignment graph embedded in a control flow graph.

1 17. The computer program product of claim 13 wherein said moving step results
2 in said second memory operation being advanced in a schedule of machine
3 instructions.

1 18. The computer program product of claim 13 wherein said first memory
2 operation is positioned before a repetitive loop in said procedure and said second
3 memory operation is within said repetitive loop; said graphic representation including
4 a phi node that corresponds to a loop back position in said repetitive loop;

5 said instructions for designating further comprising instructions for advancing
6 through said repetitive loop in order to determine a location type for each memory
7 operation in said repetitive loop; wherein
8 when a memory operation having the same location type as said first and
9 second memory operation exists in said loop, said new position in said graph
10 representation is a position that is serially dependent upon said phi node; and
11 when a memory operation having the same location type as said first and
12 second memory operation does not exist in said loop, said new position in said
13 graph representation is a position that is not serially dependent on any operation in
14 the loop.

1 19. The computer program product of claim 13 wherein said first memory
2 operation is a store or an array store and said second memory operation is a load or
3 an array load.

1 20. The computer program product of claim 13 wherein said procedure includes a
2 calling procedure and said first memory operation is in said calling procedure and
3 said second memory operation is in a called procedure that is called by an operation
4 in said calling procedure.

1 21. The computer program product of claim 20 wherein said moving step results
2 in a placement of said second memory operation in said calling procedure.

1 22. The computer program product of claim 13 wherein said location type of each
2 memory operation is selected from the group consisting of a predefined set of base
3 types, a predefined set of array types, object types, and object field types.

1 23. The computer program product of claim 13 wherein the building step further
2 comprises adding a global store dependency to each operation in said procedure
3 that reads a variable from or stores a variable to memory; the computer program
4 mechanism further comprising:

5 instructions for generating a schedule of machine instructions in accordance
6 with said graph representation, wherein each said machine instruction in said

7 schedule of machine instructions, which corresponds to an operation that reads a
8 variable from or stores a variable to memory, is ordered in accordance with said
9 global store dependency associated with said operation.

1 24. The computer program product of claim 13 wherein a first operation affects a
2 value of a variable stored in memory and a second operation serially follows said
3 first operation, said building step further comprising adding a global store
4 dependency from said second operation to said first operation; the computer
5 program mechanism further comprising:

6 instructions for generating a schedule of machine instructions in accordance
7 with said graph representation, wherein said machine instructions in said schedule
8 of machine instructions corresponding to said second operation are scheduled after
9 said machine instructions corresponding to said first operation.

1 25. A computer system for modifying serial dependencies in a procedure,
2 comprising:

3 a memory to store instructions and data;

4 a processor to execute the instructions stored in the memory;

5 the memory storing:

6 instructions for building a graph representation of said procedure, said
7 graph representation having an origin and including a unique position, relative to
8 said origin, for each memory operation in said procedure;

9 instructions for designating a location type for each memory operation
10 in said representation; each said location type based on a characteristic of said
11 corresponding memory operation;

12 instructions for identifying a first memory operation having the same
13 location type as a second memory operation; wherein said first memory operation is
14 positioned closer to said origin than said second memory operation, and said graph
15 representation does not include any additional memory operations of the same
16 location type between the first and second memory operations; and

17 instructions for determining whether said graph representation includes
18 at least one program operation between said first and second memory operations,
19 and when said determination is positive, moving said second memory operation to a

20 new position in said graph representation that is closer to said first memory
21 operation.

1 26. The computer system of claim 25 wherein:
2 the instructions for building include instructions for assigning an initial set of
3 serial dependencies between program operations represented in the graph; and
4 the instructions for moving include instructions for removing one or more of
5 the serial dependencies in said initial set of serial dependencies and creating a new
6 serial dependency from the first memory operation to the second memory operation.

1 27. The computer system of claim 25 wherein said graph representation is an
2 intermediate representation.

1 28. The computer system of claim 27 wherein said intermediate representation is
2 a static single assignment graph embedded in a control flow graph.

1 29. The computer system of claim 25 wherein said instructions for moving results
2 in said second memory operation being advanced in a schedule of machine
3 instructions.

1 30. The computer system of claim 25 wherein said first memory operation is
2 positioned before a repetitive loop in said procedure and said second memory
3 operation is within said repetitive loop; said graphic representation including a phi
4 node that corresponds to a loop back position in said repetitive loop;
5 said instructions for designating further comprising instructions for advancing
6 through said repetitive loop in order to determine a location type for each memory
7 operation in said repetitive loop; wherein
8 when a memory operation having the same location type as said first and
9 second memory operation exists in said loop, said new position in said graph
10 representation is a position that is serially dependent upon said phi node; and
11 when a memory operation having the same location type as said first and
12 second memory operation does not exist in said loop, said new position in said

13 graph representation is a position that is not serially dependent on any operation in
14 the loop.

1 31. The computer system of claim 25 wherein said first memory operation is a
2 store or an array store and said second memory operation is a load or an array load.

1 32. The computer system of claim 25 wherein said procedure includes a calling
2 procedure and said first memory operation is in said calling procedure and said
3 second memory operation is in a called procedure that is called by an operation in
4 said calling procedure.

1 33. The computer system of claim 25 wherein said instructions for moving results
2 in a placement of said second memory operation in said calling procedure.

1 34. The computer system of claim 25 wherein said location type of each memory
2 operation is selected from the group consisting of a predefined set of base types, a
3 predefined set of array types, object types, and object field types.

1 35. The computer system of claim 25 wherein the instructions for building further
2 comprises adding a global store dependency to each operation in said procedure
3 that reads a variable from or stores a variable to memory; the memory further
4 storing:

5 instructions for generating a schedule of machine instructions in accordance
6 with said graph representation, wherein each said machine instruction in said
7 schedule of machine instructions, which corresponds to an operation that reads a
8 variable from or stores a variable to memory, is ordered in accordance with said
9 global store dependency associated with said operation.

1 36. The computer system of claim 25 wherein a first operation affects a value of a
2 variable stored in memory and a second operation serially follows said first
3 operation, said building step further comprising adding a global store dependency
4 from said second operation to said first operation; the memory further storing:

5 instructions for generating a schedule of machine instructions in accordance
6 with said graph representation, wherein said machine instructions in said schedule
7 of machine instructions corresponding to said second operation are scheduled after
8 said machine instructions corresponding to said first operation.

9772-0268-999, Compaq RAD99-2751